applied sciences

MDPI

Article

On the Efficiency of OpenACC-aided GPU-Based FDTD Approach: Application to Lightning Electromagnetic Fields

Sajad Mohammadi [1], Hamidreza Karami [1,*], Mohammad Azadifar [2] and Farhad Rachidi [3]

[1] Department of Electrical Engineering, Bu-Ali Sina University, 65178 Hamedan, Iran; sajadmohamadi.el@gmail.com
[2] University of Applied Sciences of Western Switzerland (HES-SO), 1400 Yverdon-les-Bains, Switzerland; mohammad.azadifar@heig-vd.ch
[3] Electromagnetic Compatibility Laboratory, Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland; farhad.rachidi@epfl.ch
* Correspondence: hamidr.karami@basu.ac.ir; Tel.: +98-8292505-216

Received: 4 March 2020; Accepted: 23 March 2020; Published: 30 March 2020

Abstract: An open accelerator (OpenACC)-aided graphics processing unit (GPU)-based finite difference time domain (FDTD) method is presented for the first time for the 3D evaluation of lightning radiated electromagnetic fields along a complex terrain with arbitrary topography. The OpenACC directive-based programming model is used to enhance the computational performance, and the results are compared with those obtained by using a CPU-based model. It is shown that OpenACC GPUs can provide very accurate results, and they are more than 20 times faster than CPUs. The presented results support the use of OpenACC not only in relation to lightning electromagnetics problems, but also to large-scale realistic electromagnetic compatibility (EMC) applications in which computation time efficiency is a critical factor.

Keywords: graphics processing unit (GPU); OpenACC (open accelerators); finite difference time domain (FDTD); lightning magnetic fields

1. Introduction

Rigorous modeling and solution of large-scale electromagnetic compatibility (EMC) problems often require prohibitive computational resources. Fast algorithms and techniques, as well as hardware platforms with high pipelining capability, are usually used to circumvent this problem (e.g., [1–15]). Graphics processing units (GPUs) are characterized by a high parallelism capability. Figure 1 presents schematically the CPU and GPU architectures. As can be seen in Figure 1, the number of threads in GPUs is dramatically higher than that in CPUs. A CPU consists of a few immense arithmetic logic unit (ALU) cores aiming to process with high cache memory and one enormous control module capable of managing a few threads at the same time. On the other hand, a GPU includes many small ALUs, small control modules, and a small cache. Furthermore, it can execute thousands of threads concurrently since it is optimized for parallel operations [16]. One of the particular examples of a large-scale EMC problem is the evaluation of lightning electromagnetic fields, and coupling to structures. It is a large-scale problem because of (i) the scale of the solution domain, which can be in the order of several tens of kilometers, and (ii) the complexity of the propagation media, when considering the inhomogeneity and roughness of the soil (e.g., [17–24]). In most of the studies focused on the evaluation of lightning radiated electromagnetic fields and their induced disturbances on nearby structures, such as overhead transmission lines and buried cables (e.g., [25–31]), computations have been carried out by making use of CPU-based systems.

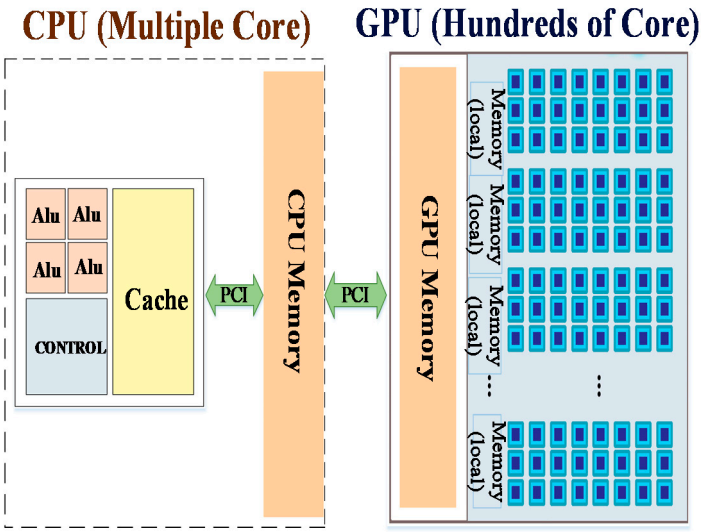

Figure 1. CPU and graphics processing unit (GPU) architectures.

However, there exist several attempts in which different hardware platforms have been used for the same study (e.g., [32–36]). One example is the use of GPU-based compute unified device architecture (CUDA) programming for the finite difference time domain (FDTD) evaluation of lightning electromagnetic fields. Due to its high-speed calculation, this approach facilitates three-dimensional modeling of a problem, taking into account parameter uncertainties such as irregular lightning channel and surface roughness [34]. It is noted that this approach is supposedly 100 times faster than the serial processing approach in CPU [33]. Although the GPU-based CUDA programming approach is highly efficient and provides the programmer with the flexibility to utilize various memories such as cache memory, it requires the involvement of the programmer for the determination of many programming details [37]. The OpenACC programming model suggested by NVIDIA, CAPS, Cray, and the Portland Group is a general user-driven directive-based parallel programming model developed for engineers and scientists in OpenACC. The programmer has the capability of incorporating compiler directives and library routines into FORTRAN, C, or C++ source codes to assign the area within the code that needs expedition in parallel on GPU. In fact, the programmer is not preoccupied with parallelism details and, in turn, leaves these tasks to the compiler. This programming model efficiently and intelligently launches the kernels and parallels the code onto the GPU. OpenACC was proposed for the first time in 2011 as a high-level programming approach that has offered almost all types of processors with high performance and portability. This programming model allows executing and creating codes using the available and future graphics hardware [38]. This is because OpenACC can transfer calculations and data from the host to the accelerator device. Despite their different architectures, the two former hardware components (host and accelerator device) and their corresponding memories are either shared or separated. An accelerated computing model of OpenACC is presented in Figure 2. According to Figure 2, the OpenACC compiler executes the code and manages the transferred data between the host and accelerator device. Furthermore, OpenACC benefits from a top-level compiler directive for the sake of portability to parallelize different sections of the code and use parallel and optimized compilers to develop and execute the corresponding codes. Several compilation directives or programs can be defined by OpenACC for parallel execution of code fragments. Recently, graphical processing using OpenACC has caught the attention of many researchers due to its relative simplicity and high performance (e.g., [39,40]). One of the particular examples of using OpenACC on graphics processors is to calculate the vector potential using the finite difference generated by time-domain Green's functions of layered media where the computational speed was 45.97 times the CPU computational speed on MATLAB [40].

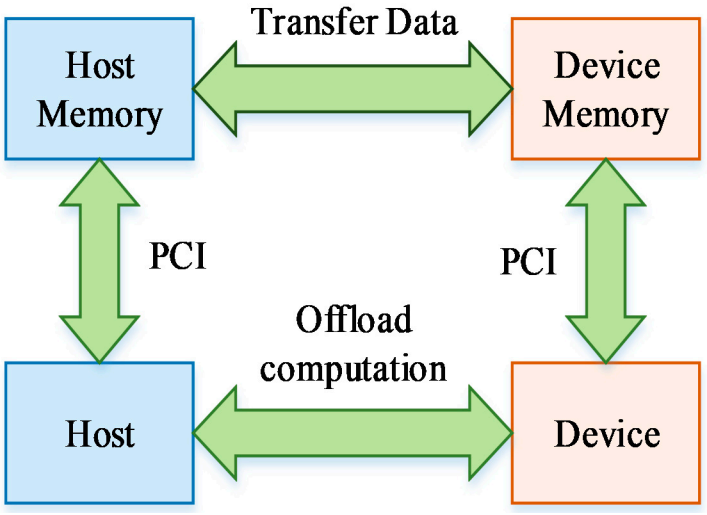

Figure 2. Open accelerator (OpenACC) model [38].

In this paper, OpenACC-based GPU processing is used to tackle the problem of long computation time in the FDTD method for the evaluation of electromagnetic fields generated by a lightning channel. It is worth noting that OpenACC benefits from relatively simple programming and dramatically increases computational speed. The rest of the paper is organized as follows. Section 2 presents the required steps for executing the computational FDTD code in graphics processing using OpenACC. Section 3 describes the adopted models and computational methods, as well as FDTD parameters. Section 4 presents the processing speed for the proposed method, and the results are compared with CPU serial processing speed. Concluding remarks are provided in Section 5.

2. OpenACC Implementation

2.1. FDTD Algorithm

The flowchart of the FDTD algorithm to calculate lightning electromagnetic fields consists of five main modules [41,42], as shown in Figure 3. In the FDTD algorithm, first, all the variables and parameters are set and defined in the initialization step, followed by the assignment of the required memory to store each component of the electromagnetic field. In each time step, the lightning channel source model excites the desired space. Then, the components of the electric (E_x, E_y, E_z) and magnetic (H_x, H_y, H_z) fields are computed and updated. Finally, boundary conditions are applied to avoid unwanted reflections of the electromagnetic field. These steps are repeated until the simulations are performed in the desired time period. The output of the FDTD algorithm would be the electric and magnetic fields at each observation point within the specified time period.

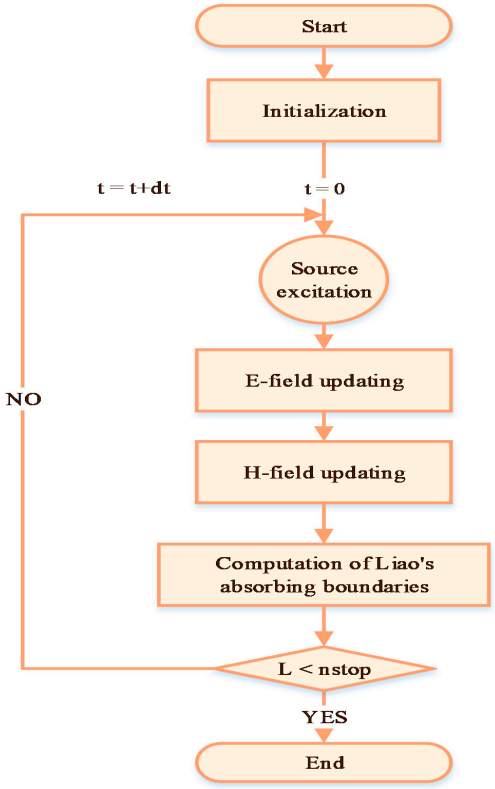

Figure 3. Finite difference time domain (FDTD) algorithm flowchart.

2.2. Hardware and Software Used

For the GPU implementation of the adopted FDTD approach, the GeForce GTX 1050 Ti hardware and PGI Community Edition 17.10 software were used. The PGI compiler was developed by the Portland Group and NVidias [43], and the PGI Community Edition includes a free license of the PGI C++ compilers and tools for multicore CPUs and NVIDIA Tesla GPUs. Community Edition enables the development of performance-portable High-performance computing (HPC) applications with a uniform source code in the most widely used parallel processors and systems [44]. Details of the GPU are presented in Table 1. For comparison purposes, we also repeated the calculations on CPU by making use of an Intel® Core™ i7-6800K @ 3.40 GHz hardware.

Table 1. GPU specifications.

Name	GeForce GTX 1050 Ti
GPU Architecture	Pascal
NVIDIA CUDA® Cores	760
Standard Memory	4 GB GDDR5
Memory Speed	7 Gbps
Graphics Clock (MHz)	1290
Processor Clock (MHz)	1392

2.3. OpenACC Programming

This section aims to provide more elaborate details about OpenACC programming, as well as compiler specifications for building and executing codes on the GPU.

2.3.1. OpenACC Data Clause

The data required for the process implementation are transferred to the GPU memory prior to the parallel region in the code, and the appropriate OpenACC directives are then inserted into the

main code for each loop. Since it is burdensome for a compiler to determine whether data will be needed in the future, they are copied conscientiously in case of future demand. Data locality helps to solve this problem so that the data remain in the local memory in the host or system as long as needed. Data clauses enable the programmer to control the method and time of generating/copying the data from/on the device. The data directives are briefly described as follows:

- Copy: Provision of space for the variables listed on the device, initialization of the variables via copying data on the device at the region's first parts, copying the generated results on the host at the region's last parts, and, finally, releasing the space on the device [38,45];
- Copyin: Provide the required space for the variables available on the device list, carry out the initialization of the variables via copying data on the device at the region's beginning, and free the space on the device without copying the data back onto the host [38,45].

In the FDTD calculations, the start and end points of data locality were positioned before the time loop (iteration) and at the end of all modules and functions, respectively. As shown in Figure 4, the starting point of data locality in the FDTD algorithm was placed before the time loop (iteration) in the code so that the proper data needed for the process execution were moved to the GPU memory using copy directive. The ending point of data locality was also placed after performing (execution) all of the modules and functions so that the generated output data returned to the CPU memory using copyin directive. This data directive avoids extra and inefficient data transfer for the calculation. Therefore, the data transfer time between the GPU and CPU is reduced using the data clause, remarkably reducing the computation time.

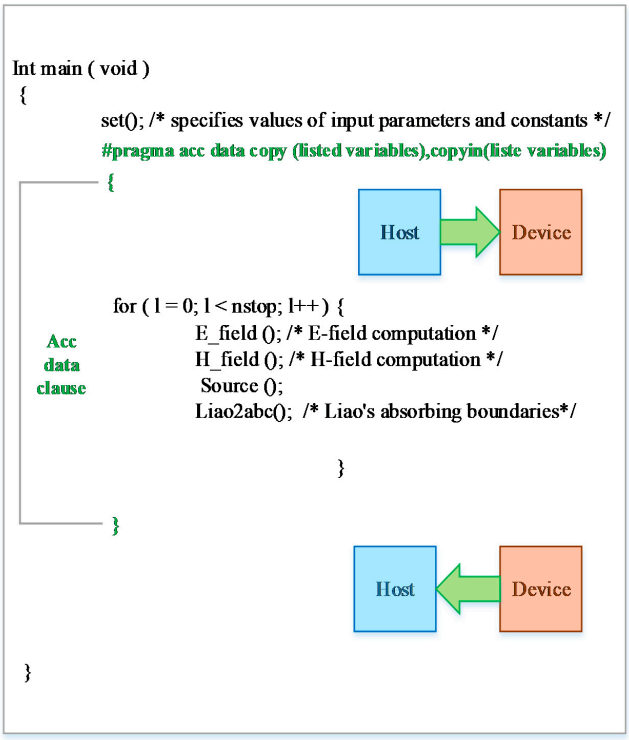

Figure 4. Optimized data locality.

2.3.2. OpenACC Parallelize Loops

Parallel and kernel regions are two different approaches to incorporate parallelism into the code provided by OpenACC:

- Structure of kernels: The structure of kernels determines a code region that can contain parallelism. However, the automatic parallelization capabilities of the compiler, identification of the loops that

are secure enough for parallelization, and acceleration of these loops influence the analysis of the region. The compiler is free to select the best way of mapping the parallelism in the loops onto the hardware [38,45];

- Structure of parallel: If this directive is put in a loop, the programmer claims that the affected parallelization loop is secured and enables the compiler to choose how to schedule the loop iterations on the target accelerator. In this case, the programmer determines the availability of parallelism per se, whereas the decision regarding mapping parallelism onto the accelerator depends on the compiler's knowledge of the device [38,45].

The OpenACC parallelizing directives were applied to different parts of the code to implement parallel computing. For example, as can be seen in Figure 5, the #pragma ACC kernels loop is set at the beginning of each of the three loops, which calculate and update each of the electromagnetic components. Table 2 presents the computation time associated with these two directives. According to Table 2, C on CPU runtime (without optimization) is 2231.59 s, which is the benchmark time. The speed was increased by a factor of 2.43 by optimizing the programming C on CPU. Moreover, as can be observed in the table, the computing speed on the GPU processor increased by factors of 10.49 and 11.14 by using different directive OpenACC including the #pragma ACC parallel loop and the #pragma ACC kernels loop OpenACC, respectively.

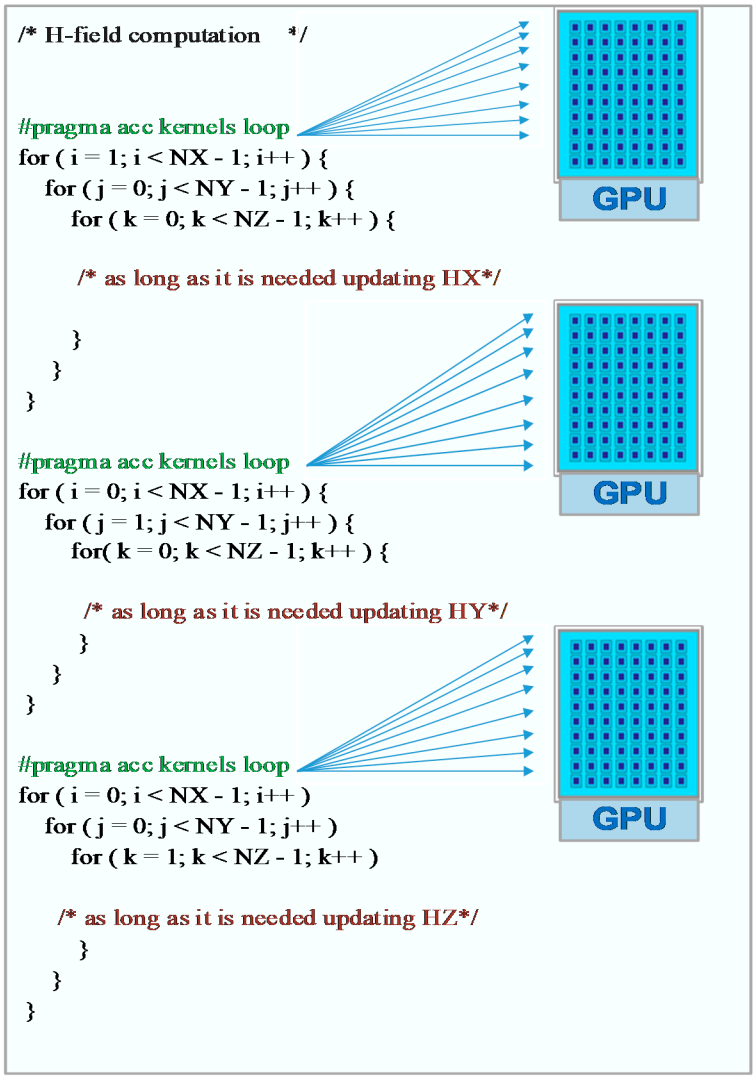

Figure 5. Mapping parallelism onto the hardware.

Table 2. Comparison of OpenACC directive.

Programming Language	Directive	Runtime(s)	Speed-Up
C (without optimization)	-	2231.59	1 X
C (with optimization)	-	914.98	2.43 X
OpenACC	#pragma ACC parallel loop #pragma ACC kernels loop	212.55 200.21	10.49 X

2.3.3. Save Variable

As shown in Figure 1, a GPU processor is presented as a set of multiprocessors, each with its own stream processors and memory. The stream processors have the full capability to execute integer and single precision floating point arithmetic, including extra cores applied to double-precision procedure [16]. The results of Table 3 reveal that when the variables are of the float type, the calculation time in the GPU process is reduced. Precisely, in GPU processing by storing variables with single precision, the computing speed can be increased up to 20.02 times the speed of storing variables by double processing in the CPU series.

Table 3. Increase in the computational speed regarding the accuracy of the variables stored.

Programming Language	Variable Precision	Runtime(s)	Speed-Up
C (without optimization)	double float	2231.59 2269.56	1 X 0.98 X
OpenACC	double float	200.51 111.47	11.01 X 20.02 X

2.3.4. Optimization by Tuning Number of Vectors

The OpenACC performance model includes three sections: Gang (Thread block), Worker (Warp), and Vector (Threads in a Warp) [38].

Gangs, vectors, and workers can be automatically set up by the OpenACC compiler or by the developer to yield the optimum approach. The number of vectors is modified in the OpenACC directives to gain the maximum occupancy and speed-up in GPU. Modifying the vector length clause can be beneficial to reaching the optimal runtime value by trying different vector lengths for the accelerator, which was used in this study. Figure 6 shows the relative runtime derived by varying the vector length compared to the compiler-selected value. It is worth noting that the best performance was obtained for a vector length of 1024.

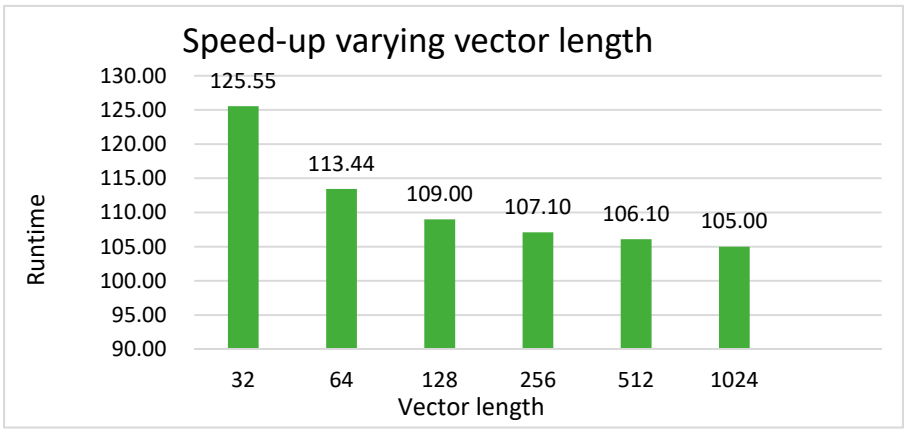

Figure 6. Relative runtime from varying vector length from the default value.

3. Application: Models and Hypotheses Considered for Analysis

The geometry of the problem is shown in Figure 7. The aim is to evaluate lightning electromagnetic fields over a mountainous terrain by making use of the FDTD technique. We assume a pyramidal mountain with a height H_m, which is varied from 0 (flat ground case) to 1000 m, while its cross-section is assumed to have an area of A = 40,000 m^2. The ground and mountain are assumed to be a perfect electric conductor (PEC) and the ground depth h_m = 100 m. The vertical electric field and the radial magnetic field components are calculated for three observation points A, B, and C. These points are, respectively, 100, 400, and 500 m far from the lightning channel base, and all located on the ground surface. In the FDTD calculations, a vertical array of current sources was used for the modeling of the lightning channel [46]. The modified transmission line model with exponential decay (MTLE) [47,48] was used for modeling the return stroke channel. For subsequent stroke current waveforms, a summation of two Heidler functions was used [49], which is given in Equation (1):

$$i(t) = \left[\frac{I_{01}}{\eta_1} \frac{\left(\frac{t}{\tau_{11}}\right)^{n_1}}{1+\left(\frac{t}{\tau_{11}}\right)^{n_1}} e^{\frac{t}{\tau_{21}}} + \frac{I_{02}}{\eta_2} \frac{\left(\frac{t}{\tau_{12}}\right)^{n2}}{1+\left(\frac{t}{\tau_{12}}\right)^{n_2}} e^{\frac{t}{\tau_{22}}} \right] \quad (1)$$

where $I_{01} = 10.7$ kA, $\tau_{11} = 0.25$ µs, $\tau_{21} = 2.5$ µs, $n_1 = 2$, $\eta_1 = 0.639$, $I_{01} = 6.5$ kA, $\tau_{11} = 0.25$ µs, $\tau_{21} = 230$ µs, $n_1 = 2$, and $\eta_1 = 0.867$, respectively [49]. The current decay constant and the return stroke speed are considered to be λ = 2000 m and $v = 1.5 \times 10^8$ m/s, respectively. The time increment was set to 9.5 ns. The dimensions of the computational domain are 1500 × 1500 × 1500 m with a spatial mesh grid composed of square cells of 5 × 5 × 5 m. In an unbounded space, the electromagnetic response analysis of a structure absorbing boundary conditions in which unwanted reflections are suppressed must be applied to planes to truncate and limit the open space and the working volume, respectively. In order to avoid reflections at the domain boundaries, Liao's condition is adopted as the absorption boundary condition [50]. The horizontal magnetic field (H_y) components at the observation points A, B, and C are calculated assuming different heights for the mountain. The results are presented in Figure 8.

According to Figure 8, the time delay, peak value, waveshape, and amplitudes of the horizontal magnetic fields (H_y) are affected by the presence of the mountain. Although the effect of the mountain on the time delay (peak field and onset time) was negligible for point A, its effect on points B and C (right side of the hill) was significant [51]. The results represented in Figure 8 have been validated using canonical structures against the simulation results obtained in [51]. The only difference was that the GPU was used and the dimensions were 1500 × 1500 × 1500 m in this study, whereas a CPU processor was used in [51]. Figure 9 presents, for validation purposes, the obtained results compared with those obtained using a 2D-FDTD method.

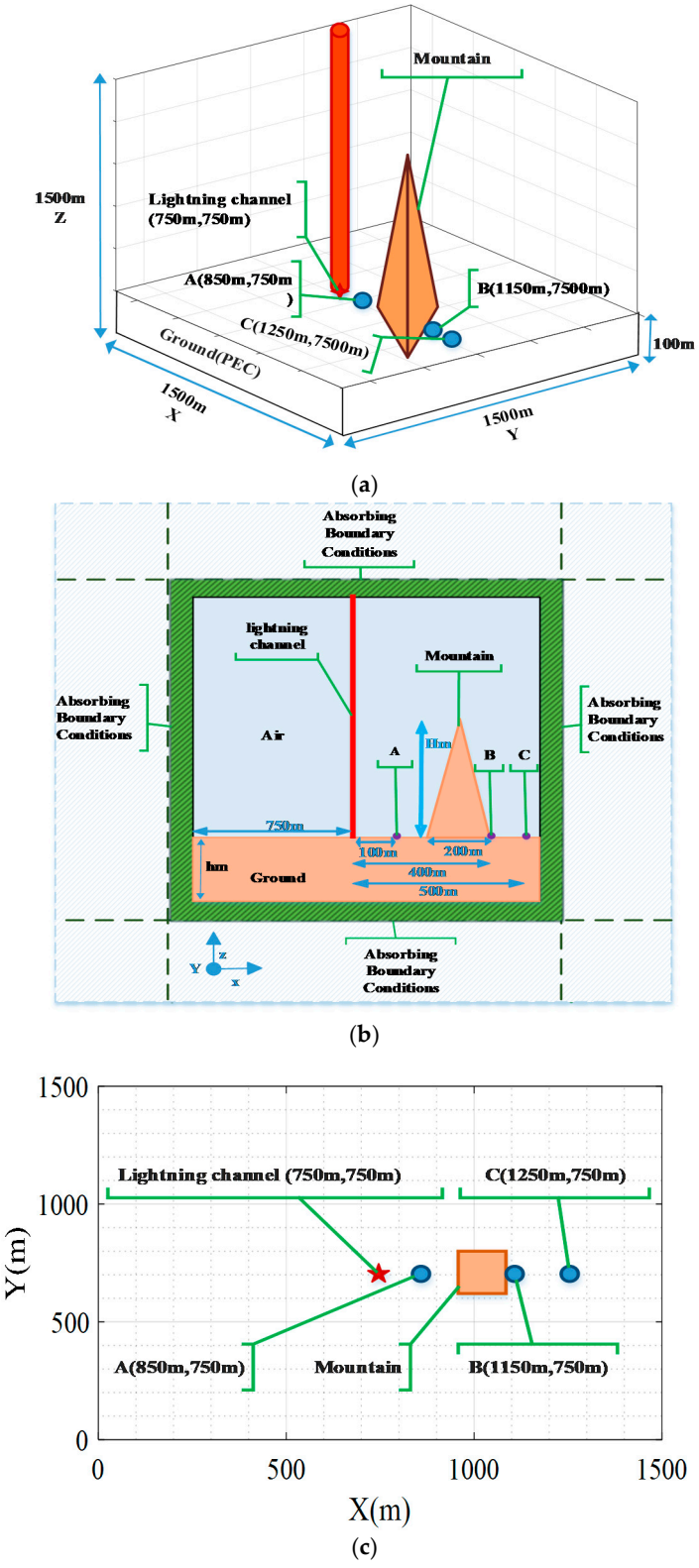

Figure 7. The simulation domain of the 3D-FDTD method and geometry for the lightning electromagnetic fields over mountainous terrain: (**a**) three-dimensional perspective, (**b**) side view, (**c**) top view.

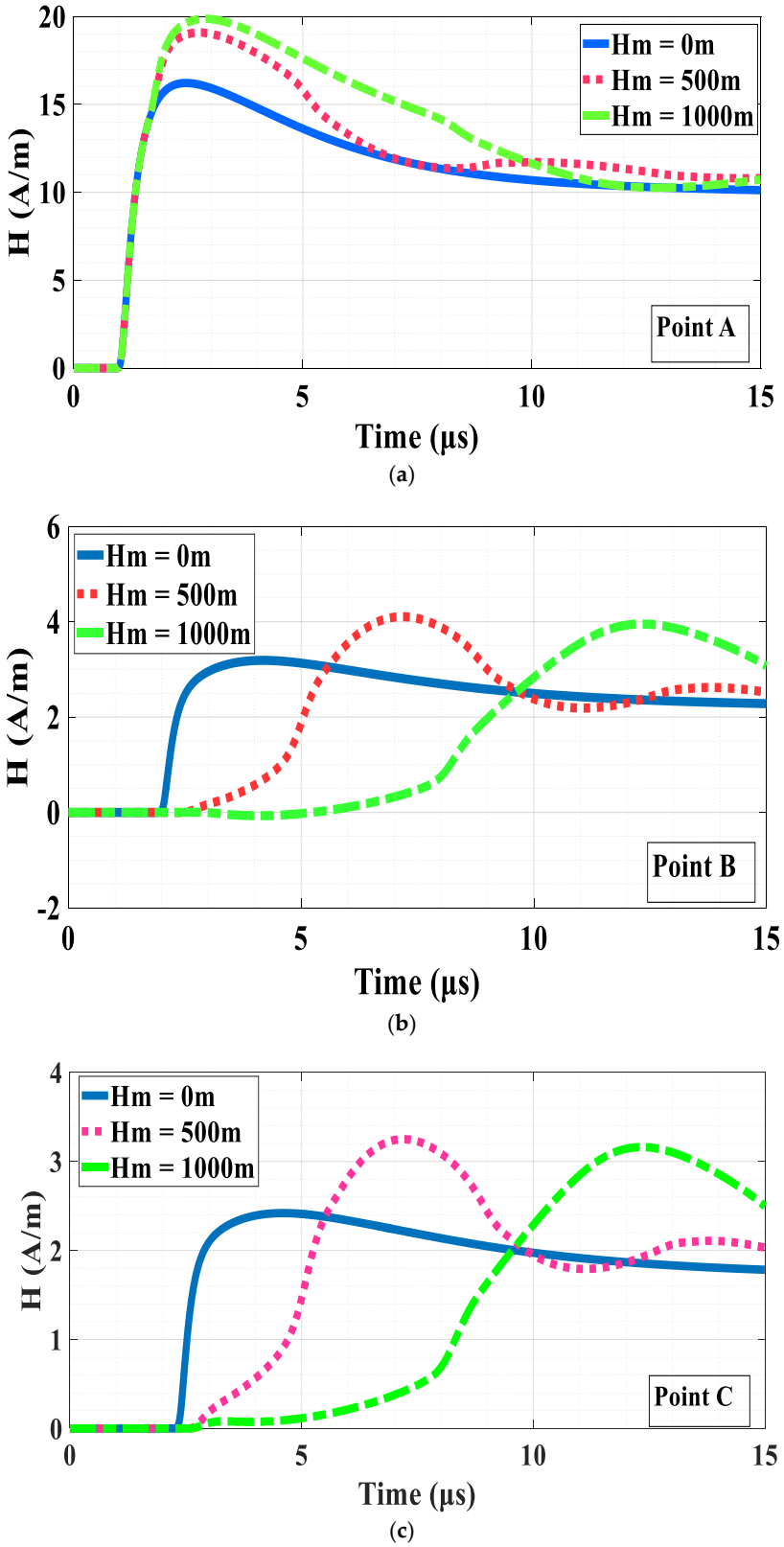

Figure 8. Horizontal magnetic field (H_y) associated with a subsequent return stroke at the considered 3 observation points A (**a**), B (**b**), C (**c**), and considering different mountain heights (H_m = 0, 200, 500, and 1000 m).

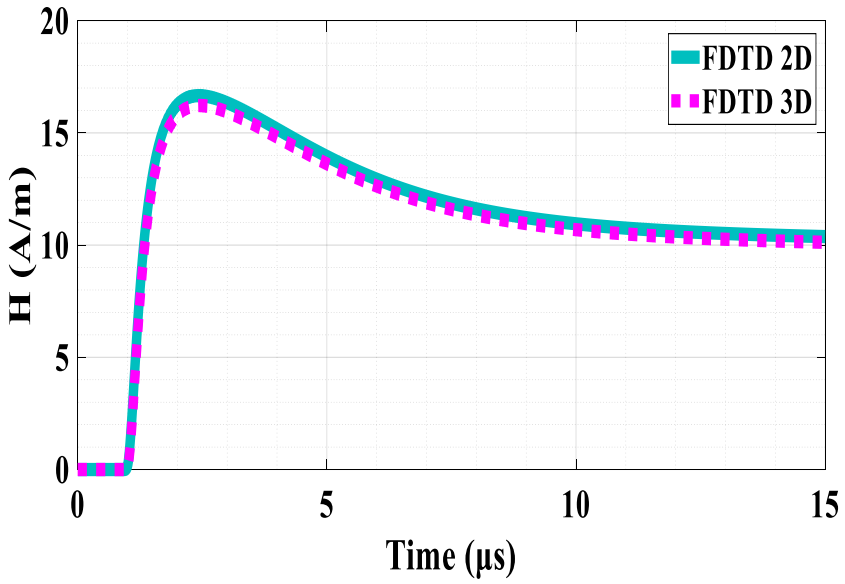

Figure 9. Horizontal magnetic field (H_y) on a smooth ideal ground at observation point A. The height of the mountain is set at H_m = 0 m. Results were calculated using the 2D-FDTD (solid line) and the 3D-FDTD (dashed line) methods.

4. Performance Analysis

This section compares the graphics processing time using OpenACC for the considered 3D-FDTD problem in this study (dimensions of the volume 1500 × 1500 × 1500 m), with the execution time of this algorithm using a serial C programming language in CPU. Table 4 presents the comparison of the performance of OpenACC compared to CPU serial C. As can be seen in Table 4, the OpenACC execution time in an efficient state was 21.22 times that of the serial processing in the CPU without compromising precision. As can be seen in Table 3, the calculation time in the GPU process was drastically reduced by storing the variables of the float type (single precision). Although float variables (single precision) led to a decrease in the accuracy of the computation, this reduction was negligible. Figure 10 depicts the difference (computing error) between CPU and GPU processing for point A, at which the mountain height was zero (H_m = 0). The error is smaller than 0.008% and can be considered insignificant. In addition, Figure 11 indicates the gain in the computation speed as a function of the size of the workspace. It can be seen that the GPU performance increases with larger simulation domains.

Table 4. Performance of OpenACC implementation.

Programming Language	Processing Time(s)	Speed-Up
CPU serial C	2231.59	1 X
OpenAcc	105.15	21.22 X

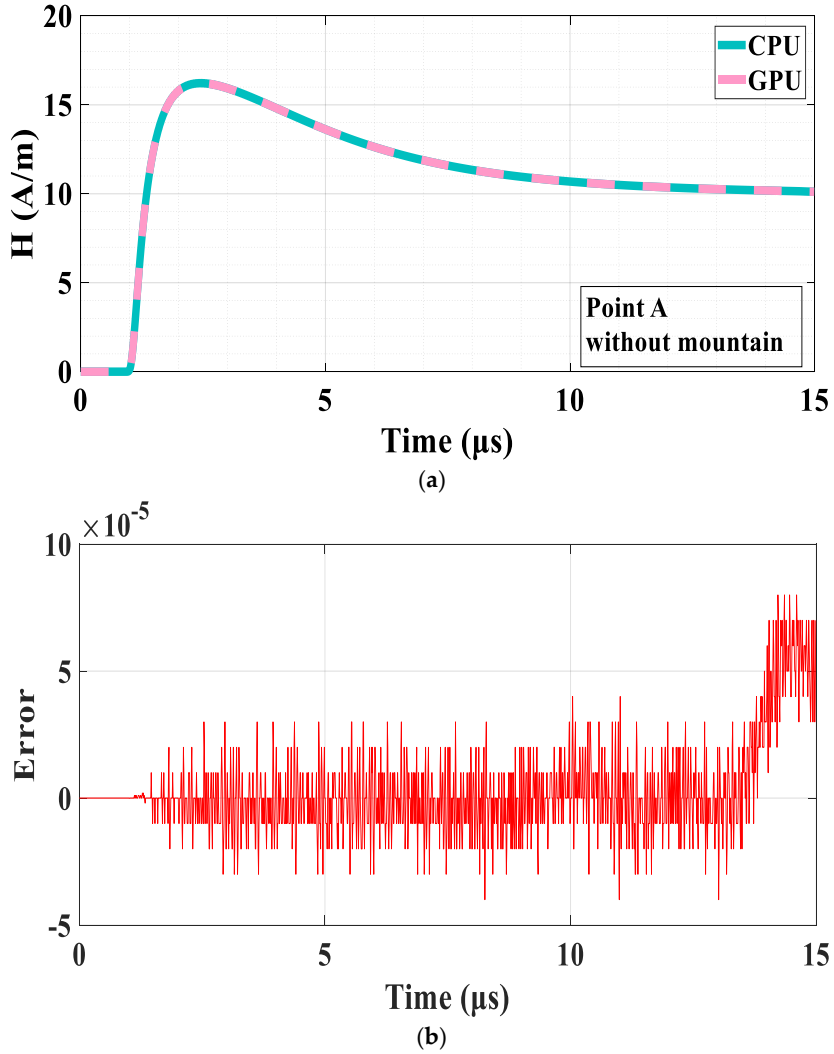

Figure 10. (**a**) The blue solid line and pink line represent the CPU and GPU processing, respectively, (**b**) Numerical error between the CPU and GPU processing.

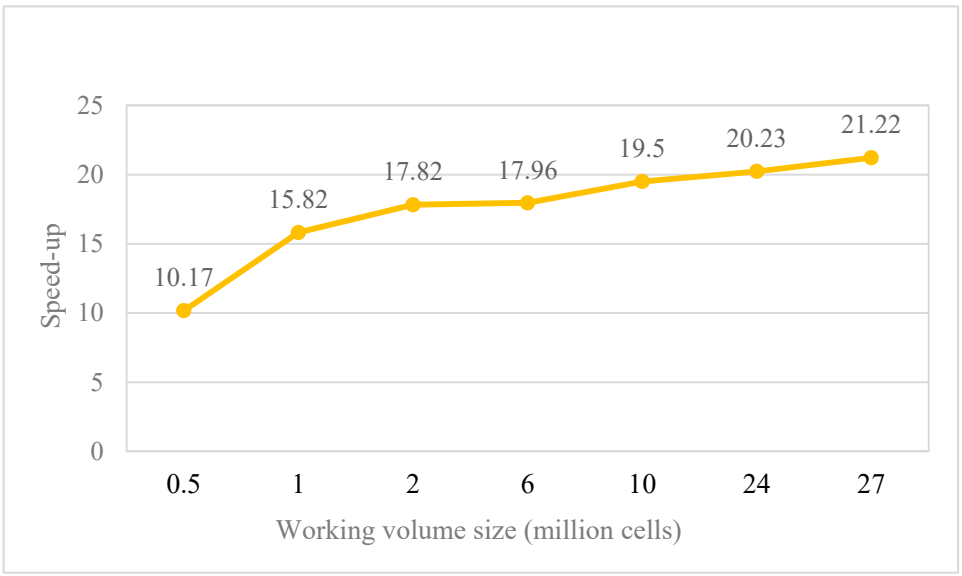

Figure 11. Gain in GPU runtime with respect to CPU for various numbers of nodes.

5. Conclusions

An OpenACC-aided graphics processing unit (GPU)-based finite difference time domain (FDTD) method was presented for the first time for the 3D evaluation of lightning radiated electromagnetic fields along a complex terrain with arbitrary topography. The OpenACC (Open accelerators) directive-based programming model was used to enhance the computational performance and the results were compared with those obtained by using a CPU-based model. It was shown that OpenACC GPUs can provide very accurate results while being more than 20 times faster than CPUs. The presented results support the use of OpenACC not only for lightning electromagnetics problems, but also for large-scale realistic EMC applications in which computation time efficiency is a critical factor.

Author Contributions: Conceptualization, H.K.; methodology, S.M and H.K.; software, S.M; validation, S.M; resources, H.K.; writing—original draft preparation, S.M.; writing—review and editing, H.K., M.A., and F.R.; supervision, H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Bakhtar Regional Electricity Company, Arak, Iran.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Balevic, A.; Rockstroh, L.; Tausendfreund, A.; Patzelt, S.; Goch, G.; Simon, S. Accelerating simulations of light scattering based on finite-difference time-domain method with general purpose GPUs. In Proceedings of the 2008 11th IEEE International Conference on Computational Science and Engineering, Sao Paulo, Brazil, 16–18 July 2008; IEEE: Piscataway, NJ, USA; pp. 327–334.
2. Bracken, E.; Polstyanko, S.; Lambert, N.; Suda, R.; Giannacopulos, D.D. Power aware parallel 3-D finite element mesh refinement performance modeling and analysis with CUDA/MPI on GPU and multi-core architecture. *IEEE Trans. Magn.* **2012**, *48*, 335–338. [CrossRef]
3. Cannon, P.D.; Honary, F. A GPU-accelerated finite-difference time-domain scheme for electromagnetic wave interaction with plasma. *IEEE Trans. Antennas Propag.* **2015**, *63*, 3042–3054. [CrossRef]
4. Chen, Y.; Zuo, S.; Zhang, Y.; Zhao, X.; Zhang, H. Large-scale parallel method of moments on CPU/MIC heterogeneous clusters. *IEEE Trans. Antennas Propag.* **2017**, *65*, 3782–3787. [CrossRef]
5. Fan, T.-Q.; Guo, L.-X.; Liu, W. A novel OpenGL-based MoM/SBR hybrid method for radiation pattern analysis of an antenna above an electrically large complicated platform. *IEEE Trans. Antennas Propag.* **2015**, *64*, 201–209. [CrossRef]
6. Guan, J.; Yan, S.; Jin, J.-M. An accurate and efficient finite element-boundary integral method with GPU acceleration for 3-D electromagnetic analysis. *IEEE Trans. Antennas Propag.* **2014**, *62*, 6325–6336. [CrossRef]
7. Ikuno, S.; Fujita, Y.; Hirokawa, Y.; Itoh, T.; Nakata, S.; Kamitani, A. Large-scale simulation of electromagnetic wave propagation using meshless time domain method with parallel processing. *IEEE Trans. Magn.* **2013**, *49*, 1613–1616. [CrossRef]
8. Kaburcuk, F.; Demir, V.; Elsherbeni, A.Z.; Arvas, E.; Mautz, J.R. Time-Domain Iterative Multiregion Technique for 3-D Scattering and Radiation Problems. *IEEE Trans. Antennas Propag.* **2016**, *64*, 1807–1817. [CrossRef]
9. Masumnia-Bisheh, K.; Forooraghi, K.; Ghaffari-Miab, M.; Furse, C.M. Geometrically Stochastic FDTD Method for Uncertainty Quantification of EM Fields and SAR in Biological Tissues. *IEEE Trans. Antennas Propag.* **2019**, *67*, 7466–7475. [CrossRef]
10. Park, S.-M.; Kim, E.-K.; Park, Y.B.; Ju, S.; Jung, K.-Y. Parallel dispersive FDTD method based on the quadratic complex rational function. *IEEE Antennas Wirel. Propag. Lett.* **2015**, *15*, 425–428. [CrossRef]
11. Sypek, P.; Dziekonski, A.; Mrozowski, M. How to render FDTD computations more effective using a graphics accelerator. *IEEE Trans. Magn.* **2009**, *45*, 1324–1327. [CrossRef]
12. Unno, M.; Aono, S.; Asai, H. GPU-based massively parallel 3-D HIE-FDTD method for high-speed electromagnetic field simulation. *IEEE Trans. Electromagn. Compat.* **2012**, *54*, 912–921. [CrossRef]
13. Wan, J.; Lei, J.; Liang, C.-H. An efficient analysis of large-scale periodic microstrip antenna arrays using the characteristic basis function method. *Prog. Electromagn. Res.* **2005**, *50*, 61–81. [CrossRef]
14. Xu, K.; Fan, Z.; Ding, D.-Z.; Chen, R.-S. GPU accelerated unconditionally stable Crank-Nicolson FDTD method for the analysis of three-dimensional microwave circuits. *Prog. Electromagn. Res.* **2010**, *102*, 381–395. [CrossRef]

15. Zainud-Deen, S.H.; Hassan, E.; Ibrahim, M.S.; Awadalla, K.H.; Botros, A.Z. Electromagnetic scattering using GPU-based finite difference frequency domain method. *Prog. Electromagn. Res.* **2009**, *16*, 351–369. [CrossRef]
16. Micikevicius, P. 3D finite difference computation on GPUs using CUDA. In Proceedings of the 2nd Workshop on General Purpose Processing on Graphics Processing Units, Washington, DC, USA, 8 March 2009; pp. 79–84.
17. Karami, H.; Rachidi, F.; Rubinstein, M. On practical implementation of electromagnetic models of lightning return-strokes. *Atmosphere* **2016**, *7*, 135. [CrossRef]
18. Li, D.; Azadifar, M.; Rachidi, F.; Rubinstein, M.; Paolone, M.; Pavanello, D.; Metz, S.; Zhang, Q.; Wang, Z. On lightning electromagnetic field propagation along an irregular terrain. *IEEE Trans. Electromagn. Compat.* **2015**, *58*, 161–171. [CrossRef]
19. Li, D.; Luque, A.; Rachidi, F.; Rubinstein, M. Evaluation of Ionospheric D region Parameter Using Simultaneously Measured Lightning Current and 380-km Distant Electric Field. In Proceedings of the AGU Fall Meeting Abstracts, Washington, DC, USA, 10–14 December 2018.
20. Li, D.; Rachidi, F.; Rubinstein, M. FDTD Modeling of lightning electromagnetic field propagation over mountainous terrain. In Proceedings of the 2019 International Applied Computational Electromagnetics Society Symposium (ACES), Miami, FL, USA, 14–19 April 2019; pp. 1–2.
21. Li, D.; Rubinstein, M.; Rachidi, F.; Diendorfer, G.; Schulz, W.; Lu, G. Location accuracy evaluation of ToA-based lightning location systems over mountainous terrain. *J. Geophys. Res. Atmos.* **2017**, *122*, 11760–11775. [CrossRef]
22. Mostajabi, A.; Li, D.; Azadifar, M.; Rachidi, F.; Rubinstein, M.; Diendorfer, G.; Schulz, W.; Pichler, H.; Rakov, V.A.; Pavanello, D. Analysis of a bipolar upward lightning flash based on simultaneous records of currents and 380-km distant electric fields. *Electr. Power Syst. Res.* **2019**, *174*, 105845. [CrossRef]
23. Oikawa, T.; Sonoda, J.; Sato, M.; Honma, N.; Ikegawa, Y. Analysis of lightning electromagnetic field on large-scale terrain model using three-dimensional MW-FDTD parallel computation. *IJTFM* **2012**, *132*, 44–50. [CrossRef]
24. Oikawa, T.; Sonoda, J.; Honma, N.; Sato, M. Analysis of Lighting Electromagnetic Field on Numerical Terrain and Urban Model using Three-Dimensional MW-FDTD Parallel Computation. *Electron. Commun. Japan* **2017**, *100*, 76–82. [CrossRef]
25. Paknahad, J.; Sheshyekani, K.; Rachidi, F. Lightning electromagnetic fields and their induced currents on buried cables. Part I: The effect of an ocean–land mixed propagation path. *IEEE Trans. Electromagn. Compat.* **2014**, *56*, 1137–1145. [CrossRef]
26. Paknahad, J.; Sheshyekani, K.; Rachidi, F.; Paolone, M. Lightning electromagnetic fields and their induced currents on buried 999bles. Part II: The effect of a horizontally stratified ground. *IEEE Trans. Electromagn. Compat.* **2014**, *56*, 1146–1154. [CrossRef]
27. Paknahad, J.; Sheshyekani, K.; Rachidi, F.; Paolone, M.; Mimouni, A. Evaluation of lightning-induced currents on cables buried in a lossy dispersive ground. *IEEE Trans. Electromagn. Compat.* **2014**, *56*, 1522–1529. [CrossRef]
28. Tatematsu, A.; Rachidi, F.; Rubinstein, M. Analysis of electromagnetic fields inside a reinforced concrete building with layered reinforcing bar due to direct and indirect lightning strikes using the FDTD method. *IEEE Trans. Electromagn. Compat.* **2015**, *57*, 405–417. [CrossRef]
29. Tatematsu, A.; Rachidi, F.; Rubinstein, M. A technique for calculating voltages induced on twisted-wire pairs using the FDTD method. *IEEE Trans. Electromagn. Compat.* **2016**, *59*, 301–304. [CrossRef]
30. Karami, H.; Sheshyekani, K.; Rachidi, F. Mixed-potential integral equation for full-wave modeling of grounding systems buried in a lossy multilayer stratified ground. *IEEE Trans. Electromagn. Compat.* **2017**, *59*, 1505–1513. [CrossRef]
31. Šunjerga, A.; Gazzana, D.S.; Poljak, D.; Karami, H.; Sheshyekani, K.; Rubinstein, M.; Rachidi, F. Tower and path-dependent voltage effects on the measurement of grounding impedance for lightning studies. *IEEE Trans. Electromagn. Compat.* **2018**, *61*, 409–418. [CrossRef]
32. Tatematsu, A.; Noda, T. Three-dimensional FDTD calculation of lightning-induced voltages on a multiphase distribution line with the lightning arresters and an overhead shielding wire. *IEEE Trans. Electromagn. Compat.* **2013**, *56*, 159–167. [CrossRef]
33. Pyrialakos, G.; Zygiridis, T.; Kantartzis, N.; Tsiboukis, T. FDTD analysis of 3D lightning problems with material uncertainties on GPU architecture. In Proceedings of the 2014 International Symposium on Electromagnetic Compatibility, Gothenburg, Sweden, 1–4 September 2014; pp. 577–582.

34. Pyrialakos, G.; Zygiridis, T.; Kantartzis, N.; Tsiboukis, T. GPU-based three-dimensional calculation of lightning-generated electromagnetic fields. In Proceedings of the 2014 International Conference on Numerical Electromagnetic Modeling and Optimization for RF, Microwave, and Terahertz Applications (NEMO), Pavia, Italy, 14–16 May 2014; pp. 1–4.
35. Pyrialakos, G.G.; Zygiridis, T.T.; Kantartzis, N.V.; Tsiboukis, T.D. GPU-based calculation of lightning-generated electromagnetic fields in 3-D problems with statistically defined uncertainties. *IEEE Trans. Electromagn. Compat.* **2015**, *57*, 1556–1567. [CrossRef]
36. Tatematsu, A. Overview of the three-dimensional FDTD-based surge simulation code VSTL REV. In Proceedings of the 2016 Asia-Pacific International Symposium on Electromagnetic Compatibility (APEMC), Shenzhen, China, 17–21 May 2016; pp. 670–672.
37. Al-Jarro, A.; Clo, A.; Bagci, H. Implementation of an explicit time domain volume integral equation solver on gpus using openacc. In Proceedings of the 29th International Review of Progress in Applied Computational Electromagnetics, Monterey, CA, USA, 24–28 March 2013.
38. OpenACC Programming and Best Practices Guide. OpenACC.org, June 2015. Available online: https://www.openacc.org/sites/default/files/inline-files/OpenACC_Programming_Guide_0.pdf (accessed on 8 November 2019).
39. Rostami, S.R.M.; Ghaffari-Miab, M. Fast computation of finite difference generated time-domain green's functions of layered media using OpenAcc on graphics processors. In Proceedings of the 2017 Iranian Conference on Electrical Engineering (ICEE), Tehran, Iran, 2–4 May 2017; pp. 1596–1599.
40. Rostami, S.R.M.; Ghaffari-Miab, M. Finite difference generated transient potentials of open-layered media by parallel computing using OPENMP, MPI, OPENACC, and CUDA. *IEEE Trans. Antennas Propag.* **2019**, *67*, 6541–6550. [CrossRef]
41. Gedney, S.D. Introduction to the finite-difference time-domain (FDTD) method for electromagnetics. *Synth. Lect. Comput. Electromagn.* **2011**, *6*, 1–250. [CrossRef]
42. Kunz, K.S.; Luebbers, R.J. *The Finite Difference Time Domain Method for Electromagnetics*; CRC press: Boca Raton, FL, USA, 1993.
43. The Portland Group—PGI. Available online: https://www.pgroup.com/about/ (accessed on 25 May 2018).
44. PGI Community Edition. Available online: https://www.pgroup.com/products/community.htm (accessed on 25 May 2018).
45. Chandrasekaran, S.; Juckeland, G. *OpenACC for Programmers: Concepts and Strategies*; Addison-Wesley Professional: Boston, MA, USA, 2017.
46. Li, D.; Zhang, Q.; Wang, Z.; Liu, T. Computation of lightning horizontal field over the two-dimensional rough ground by using the three-dimensional FDTD. *IEEE Trans. Electromagn. Compat.* **2013**, *56*, 143–148. [CrossRef]
47. Nucci, C.A. On lightning return stroke models for LEMP calculations. In Proceedings of the 19th International Conference Lightning Protection, Graz, Austria, 25–29 April 1988.
48. Rachidi, F.; Nucci, C. On the Master, Uman, Lin, Standler and the modified transmission line lightning return stroke current models. *J. Geophys. Res. Atmos.* **1990**, *95*, 20389–20393. [CrossRef]
49. Rachidi, F.; Janischewskyj, W.; Hussein, A.M.; Nucci, C.A.; Guerrieri, S.; Kordi, B.; Chang, J.-S. Current and electromagnetic field associated with lightning-return strokes to tall towers. *IEEE Trans. Electromagn. Compat.* **2001**, *43*, 356–367. [CrossRef]
50. Baba, Y.; Rakov, V.A. *Electromagnetic Computation Methods for Lightning Surge Protection Studies*; John Wiley & Sons: Hoboken, NJ, USA, 2016. [CrossRef]
51. Li, D.; Paknahad, J.; Rachidi, F.; Rubinstein, M.; Sheshyekani, K.; Zhang, Q.; Wang, Z. Propagation effects on lightning magnetic fields over hilly and mountainous terrain. In Proceedings of the 2015 IEEE International Symposium on Electromagnetic Compatibility (EMC), Dresden, Germany, 16–22 August 2015; pp. 1436–1440.